Applying Lessons from Safety-Critical Systems to Security-Critical Software

C. Warren Axelrod, Ph.D.
Decilog Inc.
Melville, New York, USA
warren.axelrod@decilog.com

***Abstract*—Researchers involved directly with the security of information-processing systems know that many such systems do not have the levels of integrity and sustainability that are much more prevalent for safety-critical systems. Safety-critical systems, many of which are industrial process control systems, are generally built and tested to much higher standards for handling system failure or aberrant behavior than is typical for even mission-critical information-processing systems.**

There is a long history of stringent standards for creating, running and sustaining safety-critical systems, particularly avionics, military systems, and the like. For example, international standard DO-178B, which was developed specifically for avionics but has been adopted by other fields, is acknowledged by the Federal Aviation Administration (FAA) and European Aviation Safety Agency (EASA) as a certification standard for avionics software. Also, NIST's Special Publication 800-82 "Guide to Industrial Control Systems (ICS) Security" provides guidance as to securing "Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC)."

In the U.S. government, the Federal Information Security Management Act of 2002 (FISMA) defines security objectives for information and information systems according to the traditional triad of confidentiality, integrity and availability. FISMA defines three levels of potential impact—low, moderate, high—on organizations or individuals were a security breach to occur.

Many private-sector organizations do in fact categorize information processing systems according to business criticality. Certain key sectors, such as financial services, are obliged to classify systems as critical in order to comply with legal and regulatory requirements. For key critical operations, systems must incorporate sufficient resiliency so as to mitigate the risk of failure. Usually criticality falls into one of three categories, namely, based on regulatory requirements or guidelines, business continuity, or class of information held, such as nonpublic personal information, and processed within the system.

In this paper we examine systems across the full spectrum of criticality, from non-critical, through security-critical and safety-critical systems, in terms of how they are engineered. That is, we look at the processes by which they are designed, built, deployed, operated, modified and decommissioned. We also discuss how mission-criticality affects the requisite level of assurance. Finally, we describe how some of the demanding methods used to strengthen safety-critical systems, which are expected to exhibit high levels of assurance and integrity, might be adapted to the engineering of security-critical information systems.

Keywords-safety-critical; security-critical; standards; software; information systems; process control systems

I. INTRODUCTION

Software security standards with respect to system and data confidentiality, integrity, and availability are very demanding in such highly-regulated sectors as banking and finance, health services, energy and transportation. Yet even systems developed and operated by, say, the most conscientious financial institutions, would not meet the stringent standards applied to the design, programming, verification, validation and deployment of safety-critical control systems, such as found in avionics and weapons systems.

It is evident from publications and the author's personal experience that there exist two distinct groups of software engineers, namely: those concentrating on security-critical information-processing systems, and those focusing on safety-critical process control systems. While there seems to be considerable information sharing among members within each group, as indicated by the volume of publications, there is relatively little communication across groups. Hardly any research and few publications appear to bridge that gap.

The goal of this paper is to show that significant benefits can be gained from fostering collaboration and information sharing and transferring good practices from those engineering safety-critical systems to others engineering security-critical systems. There are more publications about conveying good security practices from information systems to control systems than the reverse, e.g., the topic was raised in [1] and [2].

TABLE I. CATEGORIZATION OF SYSTEMS BY IMPACT OF FAILURE OR COMPROMISE

Category of System	Typical Systems	Impact of Failure or Compromise
Non-critical	Nice-to-know news systems	Minimal impact to using organization May be damaging to supplier of system
Business critical	Customer-support systems Supplier-interface systems Business information processing systems Required to be operational by law or regulation	Loss of customers Disruption of supply chain Loss of business, customer dissatisfaction Fines, removal of licenses, cessation of business
Security critical	Systems processing sensitive information, such as company secrets, non-public personal information, other sensitive information	Loss of future business due to competitors having obtained proprietary information Subject to fines by regulators Threat to national security for classified government information
Safety critical	Process control for systems upon which human safety depends	Injury and/or loss of life
Security and safety critical	Process control systems that might be subject to attack by unauthorized parties intent on doing damage	Injury and/or loss of life Reduced ability to defend the nation Delays in creation of critical infrastructure systems

II. SAFETY VS. SECURITY

A. *Definitions*

A succinct differentiation between safety-critical and security-critical software systems was given by John Barnes [3], as follows:

"Safety and security are intertwined through communication. An interesting characterization of the difference is

- Safety – the software must not harm the world,
- Security – the world must not harm the software."

In brief, the goal of safety-critical software is containment, whereas a primary aim of security-critical software is protection. This difference in outlook leads to very different criteria for ensuring safety compared to those used to ensure security. While the criteria of the two types of software do overlap to some small extent, as in the need for system integrity, they lead to very different software designs and implementations. In this paper we look at how various safety criteria and practices might be applied to security-critical software systems so that the latter take on some of the positive characteristics embodied in well-built safety-critical software systems.

B. *Criticality Criteria*

Different industries and government agencies have various criteria for criticality for their information-processing and process control systems. A single organization will usually run combinations of non-critical and mission-critical systems, the latter being security-critical, safety-critical or both, depending upon the functionality of the system and the importance of that functionality to the operation of the organization and the repercussions to the organization and others were the system to fail or operate incorrectly. In regulated industries and in government, systems are also subject to legal and regulatory criteria.

While it is not really feasible to compare the criticality of disparate systems in various industries and the public sector, the detrimental impact of some form of failure often suggests a system's criticality. For example, it is generally held that harm to humans is the most negative factor, so that safety-critical control systems, the failure or improper performance of which would lead to injury or death, are usually considered to be the most critical. With security-critical information systems, criticality is often based on the ability of the organization to continue in business, were the system to fail or the level of losses occurred because of downtime. However, the impact of loss of functional or data integrity and the leakage of sensitive data are critical in, for example, the financial services industry, as loss of confidence, though less tangible and not readily measured, can lead to business failure through defection of customers, say. Nevertheless, it is helpful to categorize systems from noncritical to extremely critical, as shown in Table 1.

C. *Security of Safety-Critical Systems*

Rather than overlaying safety-critical systems' practices on top of those from security-critical systems, we apply tools and methods derived from safety-critical software engineering practices to security-critical systems to the extent that it is practically and economically feasible to do so. The areas that offer the most opportunities for improvement, in the opinion of the author, are those related to programming languages and software assurance. However, other aspects of software systems, such as design, development and implementation, will be considered also.

The use of computer programming languages specifically tailored to safety-critical systems and implementation of formal verification processes would go a long way in improving software quality for systems that are not considered safety-critical [3].

III. Safety and Security Standards

We now examine some specific standards applied to mission-critical and safety-critical systems that would be beneficial to the resiliency and system integrity of security-critical systems.

In the author's experience, the setting of, and adherence to, standards appear to be much more stringent and better enforced for safety-critical systems that for security-critical systems. This is understandable since injury or loss of life is so much more detrimental than loss of assets. Having been involved in several efforts to establish information security standards, the author is not aware of any security standards, even a *de facto* standard such as the PCI DSS (Payment Card Industry Data Security Standard), developed for the payment card industry, which come close to avionics safety standard DO-178B [4], military standard MIL-STD-498, and IEC 61508. The security "standards" that originated with BS (British Standard) 7799, was virtually duplicated in ISO 17799, and have evolved into ISO 27001 and 27002, are much more general in nature than are the above-mentioned safety standards, and do not lead directly to equivalent levels of security or safety

A couple of examples of categorization of system types by failure and system integrity are shown in the Appendix for aircraft and motor vehicle systems. Of particular note is that there is a distinct difference between process control systems and information systems, even when deployed in the same vehicle, where the former systems are generally more safety-critical.

We now proceed through the various phases of the development and implementation lifecycle, suggesting where safety standards and practices might be useful in enhancing security-critical systems.

A. System Requirements and Design

Omissions of security requirements in the conceptual stage of creating a system generally lead to insecure systems since, clearly, if security attributes are not added as requirements and not specified in the design stage, they will not be included and will have to be bolted on once the security deficiencies are recognized, usually in the final stages of development or after the system has been implementation and is running in production.

Safety standards demand early recognition of safety requirements so that there is a very good chance that the final product will satisfy safety requirements for particular categories of system.

It is recommended that security features be added at the requirements and specifications stages of developing information systems and that information-security specialists should be included in the team creating requirements. This is not generally done in the commercial world, but would be expected to lead to much more secure systems.

B. System Architecture

In a sense, safety-critical systems have been shielded from the more egregious security attacks, until recently, because their architectures usually called for standalone, isolated systems with no connections to the outside world. That aspect is undergoing rapid change as control systems are being hooked up to networks, which are often vulnerable to attack, and linked to other systems that are not necessarily secure.

While there are many protective and defensive methods for securing systems and networks, such as firewalls, intrusion detection and prevention systems, antivirus and anti-malware products, and the like, the threats and successful exploits always are ahead of the defenses.

From time to time, highly-confidential information processing systems are isolated both physically and electronically, that is not commonplace, particularly in the commercial world. While the author recognizes that the trends towards interconnectivity and interoperability are immutable, it is suggested that, prior to connecting security-critical (and safety-critical) systems to networks and other systems, a careful analysis be done to determine whether there might be ways to eliminate or limit connectivity or create air gaps so that such systems are not subject to outside attacks. Again, it is realized that avoiding the issue in this way is not usually feasible, there are undoubtedly cases where isolation or air gaps are feasible, but are often not considered to be an option.

C. Computer Languages

There has been, and continues to be, a tradeoff between adding features to programming languages, which in turn result in fuller-featured software systems, and the securing of those systems. Successful attacks on software systems regularly exploit what are often the most appealing features of the software, such as those that perform messaging and collaboration. Easy communication often facilitates exploitation of those features for those with illegal and evil intentions. It is significant that so-called "system hardening" is frequently achieved by stripping features from an application or operating system.

There is an ongoing debate as to whether certain languages are more appropriate for safety-critical systems. Reference [3] claims that Ada, a programming language developed in 1970s at the behest of the US Department of Defense, leads to inherently more secure applications than do C, C++ and Java. As an example, Ada does not permit some of the most common vulnerabilities from other languages, such as buffer overflow conditions. There is even a stripped-down version of Ada, called SPARK, that is used for the most mission-critical or safety-critical systems. Reference [4], on the other hand, asserts that such claims are overblown and that safety-critical

systems can be written in C or C++ if appropriate care is taken. Nevertheless, the point is clear that programming languages that force discipline on the developer are more likely to produce more secure code.

D. Platforms and Infrastructure

As with applications, there are hardened versions of operating systems, such as Linux, that are used to support security-critical and safety-critical systems. For example, a firewall or SIEM (Security Information and Event Management) system will likely run on a hardened operating system as do many applications that so not require the wealth of features but do need to be resistant to attack.

Likewise, the architectures that support such systems will include a series of preventative tools, such as network and host intrusion detection and prevention systems. Also, there is a good argument to isolate really critical systems behind electronic air gaps and secured facilities.

E. Coding Practices

There is much written about how best to write secure code and test for insecure code. Largely, this is a matter of training designers and developers in secure coding practices, and then having procedures in place, such as peer code reviews and automated static testing, to catch any lapses. Since these incremental activities increase the resources needed and the time to completion, it is important to have management buy into the additional costs and time for ensuring that systems will stand up to operational events and attacks.

It is more common in the development safety-critical systems to invoke substantial review and testing processes since a system failure might have horrendous outcomes. The costs that result from failures of security-critical systems are often not as clearly defined and spread across a larger user base, so that convincing management to put in the extra effort is difficult in good times and well-nigh impossible in difficult economic times.

F. Verification and Validation (V&V)

Validation is defined as "ensuring that the right system has been implemented," and comprises inspections and reviews and obtaining appropriate sign-offs on requirements, running tests on completed programs, and the like. On the other hand, verification is defined as "ensuring that the system has been correctly implemented," and involves testing applications against requirements.

Reference [5] expresses it well, as follows: "[The implementation verification] process has the goal of ensuring that the system implementation satisfies the validated requirements. The main goal [of implementation verification is to demonstrate that the system performs all the functions and only the functions for which it has been devised." The concept of demonstrating that the system does not perform unwanted functions covers functional security testing [6].

Most texts on software engineering mention V&V as a component of the SDLC, and the particular use of the term is more common for government systems and process control systems. For commercial systems, the more common expressions used are testing and quality assurance. While clearly there is some commonality among the terms, the formal aspect engendered by the explicit inclusion of both verification and validation produces a tighter and more desirable process. Consequently, it is suggested that V&V be applied to security-critical information systems, as is done more commonly in the safety-critical systems arena.

G. Deployment

Generally, commercial information systems are installed on a number of platforms operating in a variety of infrastructures, whereas process control systems often require very specific platforms and are restricted to carefully-defined infrastructures. There are arguments against both diversity and monocultures. Diversity requires greater administrative and support effort because of having to deal with many vendors and systems. Monocultures are arguably more susceptible to their all being impacted by a single attack.

Perhaps the ideal is somewhere in between. Security-critical information systems could well benefit from a restricted number of implementations as security professionals and vendors can concentrate resources and attention on a lesser number of installation types.

H. Operations and Change Management

One can argue that system administrators and operators running process control systems are much more familiar, in general, with the functionality of the systems that they run and support, due in large part to a limited functional scope and the long operational life cycles, often with 10 -15 year periodicity. On the other hand, system administrators and operators of information systems are generally less familiar with the operation and functionality of the applications that they run and support as they are often numerous and change frequently. An exception might be highly-customized proprietary systems which require significant operator understanding and intervention and which might remain fairly stable over many years.

System	Control or Information	Level A (Catastrophic)	Level B (Hazardous)	Level C (Major)	Level D (Minor)	Level E (None)
Flight control system	Control	X				
Cockpit display and controls	Control	X				
Flight management system	Control	X				
Brakes & ground guidance system	Control		X			
Centralized alarms management	Information			X		
Cabin management system	Information				X	
Onboard communications system	Information				X	
Centralized maintenance system	Information				X	
Entertainment system	Information					X

TABLE II. RTCA/DO-178B STANDARD APPLIED TO AIRCRAFT CERTIFICATION

The lesson here is to increase training of operational and support staff in the functionality of the information systems that they run and for management to resist the pressure to frequently update current systems and introduce new ones. While it is recognized that business needs will often require somewhat frequent changes in order to remain competitive, there is a trade off to be made against the risk of lack of knowledge, which might facilitate successful attacks.

I. Decommissioning

A common difference between information systems and process control systems is that the former may well handle and contain sensitive information, whereas the latter are more likely to contain proprietary processes. Furthermore, the turnover of information systems is often more frequent than for process control systems. Consequently, the nature of the proprietary information and the low frequency of change tend to favor process control systems in terms of risks relating to decommissioning. This suggests that there are inherent factors that work against information systems. Nevertheless, it does point to the benefits of not storing sensitive data in applications themselves, so that the data are not at risk when the application is retired, and also in reducing the frequency of decommissioning systems, subject again to the business needs. It would seem to make sense to not react to every new version of software, unless there are real security benefits to upgrading.

IV. CONCLUSIONS

There are approaches used for safety-critical process control systems that would benefit security-critical information systems were they to be applied. Introducing some, such as procedural practices, is usually more realistic than introducing others, such as changing the programming languages used. Nevertheless they should all be considered to some degree.

Other aspects, such as system architecture, may be inherent in the type of system and so are less likely to benefit from cross-pollination.

APPENDIX: APPLICATION OF STANDARDS TO AIRCRAFT AND AUTOMOTIVE SYSTEMS

Reference [7] attributes various degrees of failure to aircraft (from DO-178B) and automotive vehicles (from IEC 61508) to different systems as noted in Tables II and III respectively. It is interesting to note that failures of control systems generally have much greater impact on the outside world than do information systems failures. Nevertheless, misleading information from an information system can lead to serious negative consequences if it results in the misuse of a control system. In the case of the Stuxnet computer worm's portfolio of payloads, a key attribute was the worm's ability to present fake displays of data on the performance of the subject control systems to system administrators, leading them to believe that the systems were operating correctly even though they were in self-destruct mode.

TABLE III. IEC 61508 APPLIED TO AUTOMOTIVE VEHICLES

System	Control or Info	SIL 3	SIL 2	SIL 1	N/A
Steer-by-wire	Control	X			
Brake-by-wire	Control	X			
Engine management system	Control		X		
Dashboard	Information			X	
Body controller	Information			X	
Navigation	Information				X
Diagnostic	Information				X
Entertainment system	Information				X

SIL = Safety Integrity Level (see [8] for definitions)

REFERENCES

[1] P. G. Gutgarts and A. Termin, "Security-Critical versus Safety-Critical Software," Proc. of the IEEE Homeland Security Technology Conference, Waltham, MA, pp. 507–511, November 2010.

[2] Special Issue on "Process Control Security" of IEEE Security and Privacy Journal, vol. 6, no. 6, November/December 2008.

[3] J. G. P. Barnes. "Ada," in Avionics: Elements, Software and Functions, C. R. Spitzer, Ed. Boca Raton: FL: CRC Press, 2007, pp. 15-1 to 15.50.

[4] T. K. Ferrell and U.D. Ferrell, " RTCA DO-178B/EUROCAE ED-12B," in Avionics: Elements, Software and Functions, C. R. Spitzer, Ed. Boca Raton: FL: CRC Press, 2007, pp. 16-1 to 16.11.

[5] J. H. Robb, "Hey – C and C++ can be used in safety critical applications too!" Software Tech, vol. 13, no. 1, March 2010.

[6] M. Bozzano and A. Villafiorita. Design and Safety Assessments of Critical Systems, Boca Raton, FL: CRC Press, 2011.

[7] D. Blondin, "Would certification become mandatory in automotive engineering?" ERTS Second European Conference, 2004.

[8] http://en.wikipedia.org/wiki/Safety_Integrity_Level

Dr. C. Warren Axelrod is responsible for business development and marketing at Decilog, Inc., a small-business defense contractor headquartered in Melville, New York. His career to date has been predominantly in information technology, cyber security and privacy in the financial services industry.

He represented the banking and finance sector at the Y2K command center in Washington, DC during the century date rollover. He testified at a Congressional Hearing on cyber security in 2001. He is a member of the Cloud Security Alliance and co-led the working group responsible for application security, portability and interoperability.

Dr. Axelrod received ISACA's 2009 Michael Cangemi Best Book/Best Article Award for his article "Accounting for Value and Uncertainty in Security Metrics." He was honored with the prestigious Information Security Executive (ISE) Luminary Leadership Award in 2007 and received the ***Computerworld*** Premier 100 IT Leaders Award in 2003.

Dr. Axelrod has written three books, two of which are on computer management, as well as numerous articles on a variety of information technology and information security topics. His third book is ***Outsourcing Information Security***, published in 2004 by Artech House. He presents regularly at conferences and seminars.

He holds a Ph.D. in managerial economics from Cornell University, an M.A. in economics and statistics and a B.Sc. in electrical engineering from the University of Glasgow. He is certified as a CISSP and CISM and is a member of IEEE, ACM, ISACA, and ISSA.